1-1-2005

Shapes reconstruction from robot tactile sensing

Liangchuan Mi
Iowa State University

Shapes reconstruction from robot tactile sensing

by

Liangchuan Mi

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Yan-Bin Jia, Major Professor
David Fernandez-Baca
Greg R. Luecke

Iowa State University

Ames, Iowa

2005

Graduate College
Iowa State University

This is to certify that the master's thesis of

Liangchuan Mi

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

ABSTRACT

Shapes reconstruction bridges real objects and their computer models. Most of the shape reconstruction techniques were derived for computer vision applications. A very important sense of human, tactile sensing can be applied to acquire shape information about 2D and 3D objects. Nevertheless, tactile data usually has a lot of noise. In this thesis, I present an applicable scheme that acquires shape data using a simple joystick sensor and then reconstructs 2D shapes and 3D patches. The 2D shapes are tracked by an Adept Cobra robot and represented as polynomial functions determined by the 3L fitting algorithm. The 3D shapes are composed of multiple patches, each of which is described by a polynomial function generated by least-square fitting. Experiments have been carried out with the robot. A display environment for 3D objects has also been developed.

CHAPTER 1. INTRODUCTION

2D/3D shapes are very common objects for modern computing and manipulations. Shape reconstruction is very important as a bridge between an object in the real world and its efficient representation model in computer for further use. Medical imaging, robotics manipulation, geology, microscopy and aerospace manufacturing all often employ the techniques of shape reconstruction. Those shape reconstruction techniques are highly related with computer vision, computational geometry, computer graphics, control theory, robotics manipulation, sensor techniques, etc.

The basic procedures include:

1. Input discrete sampling data of a shape into computer
2. Process these data to reconstruct a model in computer
3. Further use with the reconstruction representation of the shape

The most common way to acquire shape information of an object is using digital cameras, range sensors or other similar techniques (e.g. CT in medical diagnosis). The input data to the computer are images. The next step is to apply image processing algorithms on these images. Normally, for 2D models, algorithms will process still images or a sequence of images (a period of video). For 3D models, such algorithms will try to construct a 3D object with 2D cross section images. The presentation could be a computer graphics object (e.g. a set of line segments, basic curves, or triangles of a model), a continuous function or a set of functions. The last two methods are more difficult on computation and algorithm development. These computer vision related methods always require a lot of raw data to describe the whole shape and meantime the precision is limited by the images' resolution.

As a very important sensing of human, tactile sensing could be a method to detect the information of the shapes of 2D and 3D objects. But compared to computer vision, it is seldom used. The reasons include lack of powerful point/array tactile sensors, limited precision of existing sensors, very limited amount of data acquired, very limited sampling speed and complex control system for sensor detecting, etc. In one word, tactile sensors are not as powerful and fast as image sensors. Thus, it is a big challenge to develop an approach based on tactile sensing for 2D/3D shape reconstruction.

In my thesis, I present an applicable scheme to acquire data with a simple joystick tactile sensor and then reconstruct common 2D shapes and 3D surface patches by deriving their descriptive polynomial functions. The whole thesis will be on the whole presented in two main parts, one part is for 2D shape and the other for 3D shape patches.

Chapter 1 is overview and introduction to the whole scheme. Chapter 2 cover 2D object tracking with Cobra industrial robot, post-processing and reconstruction to closed form polynomial functions. Chapter 3 introduces some basic concepts of surface patches for 3D object reconstruction, a sampling strategy with a simple tactile sensor and reconstruction procedures for a single surface patch. Chapter 4 is a summary of the thesis and discusses future work.

1.1 Related Work

1.1.1 2D Tracking

Hybrid control has been used for generating constrained motion. The scheme, proposed by Raibert and Craig [12] applies independent position control and force control along unconstrained and constrained directions, respectively. Their work was later extended by Khatib and Burdick [8] to incorporate dynamic coupling effects. Mason [10] synthesized control strategies for compliant motions by looking into the semantics of motion primitives.

In practice, contours (and motion constraints) are often unknown. Dynamics-based tracking causes force oscillations due to joint transmissions. Such effects were compensated by Jattaetal [6] through the addition of a normal velocity feedback loop. Yoshikawa and Sudou [14] estimated the instantaneous contact frame (and thus local geometry) using the latest trajectory information and contact force measurements. Nevertheless, their frame estimation was not quite accurate since only two data points on the trajectory were used. In Section 3.1, we will describe more accurate contact geometry estimation through fitting over local data.

Baeten and De Schutter [2] employed a vision system to guide sharp corner turns, while applying force control to perform the rest of the tracking. Xiao [13] also used visual guidance fused with hybrid position/force control to follow a trajectory (specified in the image plane) on an unknown surface. Without resorting to direct force control, Lange and Herzinger [9] enhanced second round contour tracking by transforming readings from an external F/T sensor into joint torques for position control.

Fearing [5] described how a cylindrical tactile fingertip could recover the pose of a generalized convex cone from a small amount of data. Allen and Michelman [1] employed a Utah-MIT hand to obtain sparse contact points around an object and then fit a superquadric surface to the data as the reconstructed shape. Ellis and Qin [4] studied shape recovery from the strain on a tactile sensor, formulating it as an optimization problem solvable by the Levenberg-Marquardt method. Moll and Erdmann [11] showed how to simultaneously estimate the shape and motion of an unknown convex object from tactile readings on multiple manipulating palms under frictionless contact.

In the coauthor's recent work [7], a 2-axis force/torque sensor was designed to sense contacts and localize a jaw on a 2D curved shape through rolling. The sensor is, however, not accurate enough for sensing the global shape.

1.1.2 3D Patches

We had planned to move to 3D objects for a long time. The idea of "three sample curves with one intersection" is originally from Dr. Yan-Bin Jia. In real experiments, I found the effective area was quite limited, then I thought about the idea of "patches". The way we did 3D surface modeling is quite different from existing methods. Detailed information will be provided in Chapter 3.

CHAPTER 2. RECONSTRUCTION OF 2D BOUNDARY

Most shape reconstruction methods are based on vision and image processing. In this paper we describe a system that tracks unknown shapes with a joystick sensor mounted on an Adept SCARA robot and then reconstruct it in computer. The joysticks limited force sensing is compensated by the Adepts high positional accuracy to get precise contact measurements. Force control is realized by a simple feedback loop. Due to random sensor noise, the tangential motion direction from force reading alone is unreliable. A position control method is used to update the tracking direction based on a cubic fit to local turning direction of the contour. The result is fast tracking with hardly loss of shape accuracy.

2.1 Introduction

Most shape reconstruction methods are based on vision and image processing. In this paper we describe a system that tracks unknown shapes with a joystick sensor mounted on an Adept SCARA robot and then reconstruct it in computer. In the task, the robot is holding a tool to follow the contour of an object whose shape is unknown. Applications include part polishing, inspection, paint spraying, cleaning, modeling, etc. During the tracking, the tool is constrained on the surface to maintain contact while moving along the contour. Dynamics-based hybrid control needs to deal with force oscillations, which adds to the complexity of implementation. For industrial robots, direct control to the joint of the arm is not allowed. This presents an obstacle of implementation of contour tracking through force control on such robots.

In this paper, we investigate contour tracking in the context of shape reconstruction. With a robot of high positional precision, we can control the tracking motion based on contact force information only. A laser range sensor can generate the global shape more efficiently, but it is easy to miss some details. The quality of range data also deteriorates in case of motion. Meanwhile, tracking is natural in a task like manipulation when the robot hand needs to determine the local shape geometry before executing the next motion. Our tracking tool is a joystick sensor mounted on an Adept Cobra 600 robot (which has 0.02mm positional error). The movement of the sensor along the shape is controlled by a simple feedback loop. Due to random noise and contact friction, force readings from the joystick have errors as well as direction. So we cannot rely on them entirely to carry out position control. Given the Adept robots high precision, the contact precision can be limited within a very small error range. Thus a polynomial fit to a recent sequence of tracking directions should approximate the local shape very well. We then compute the tangent to the polynomial at the current contact. Combining this tangent with the direction orthogonal to that of the current force reading will generate a more reliable direction for the next step tracking motion. The above ability to estimate curvature while tracking also leads to control based on the tracking speed. Around a corner, tracking needs to slow down to capture the large variation in local geometry. In early work, such control can hardly be provided by a vision system.

Methods section describes the experimental platform and system control architecture and also focuses on the generation of tracking motion which makes use of fitting to local geometry, including post contour adjustment from force readings. Some tracking results are then presented in Result section. Conclusion section concludes with a discussion and future work.

2.2 Methods

2.2.1 System Overview

Our tracking system is implemented with an Adept Cobra robot of 4-DOF and a position control interface implemented by TCP/IP communication. The robot communicates with a host computer via a TCP/IP port. A joystick sensor from Interlink Inc. is used to obtain force readings. As shown in Figure 2.1 below:

The sensor is mounted upside down on the Adept's end of the arm. It has a force range of 20-170g and a serial interface to the host computer for data transfer. The sensor's x- and y-axes are aligned with the world coordinates so that force and position measurements have the same reference.

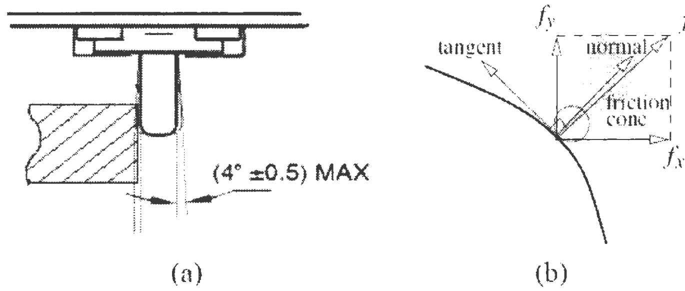


Figure 2.1 Joystick sensor and its force.

Suppose the sensor reads a force measurement $f = (f_x, f_y)$. Then the outward normal and the tangent at the contact may be estimated as $(f_x/|f|, f_y/|f|)$ and $(-f_y/|f|, f_x/|f|)$, respectively. However, force measurements by the sensor have random errors up to 10% in both x- and y-directions. Also, due to the friction, f may lie anywhere inside the contact friction cone (see Figure 2.1(b)). Later, we will describe a more accurate tangent estimation method which combines the force measurement with recent tracking history. Tangent estimation acts as a major component of the system: motion generation. Another important component is contour adjustment, as shown in Figure 2.2.

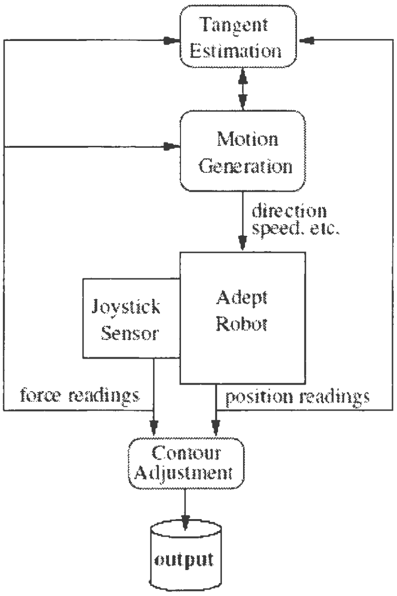


Figure 2.2 Tracking system architecture.

2.2.2 Motion Control

The motion generation module controls the direction and speed for the next sensor movement during the tracking. Based on the sensor force readings and the estimated tangent direction, one of the following three modes of motion is executed:

1. When the contact force lies within the normal range, the robot continues moving along the current tangent direction.

2. When the contact force is below the range, the joystick is breaking the contact. A motion is generated to move the joystick back toward the shape. After the contact force returns to the normal range, the tangent estimation module is trigged to compute a new tangent direction, in which the robot begins new movement.

3. When the contact force is above the range, the joystick is "pushing into" the contour. This usually happens while tracking a concavity. The joystick at first backs away from the shape until the contact force falls within the range. Then a new tangent direction is estimated for the next movement same as the second mode.

Contact measurements are taken at a rate of 10-100Hz depending on the tracking speed.

2.2.2.1 Tangent Direction

During most time of tracking, the sensor is moving along the tangent direction. The tangent direction is updated every time the measured contact force drops below a certain threshold. After the update, the sensor will continue moving along the new tangent and sample shape data at a fixed rate. Inaccurate estimation of the tangent could result in a significant number of movements in the normal direction in order to adjust the contact force, and slows down the tracking speed. Improvement on tangent estimation will reduce zigzag of the movement and thus increase the tracking speed. It will also improve more accurate shape approximation. Although boundary point measurements have individual errors, the contour still approximates the original shape very well, due to the Adept's high positional accuracy. This prompts us to fit the inaccuracy of numerical differentiation to obtain the tangent direction. The data set for fitting includes the points where the k most recent tangent updates. These points are stored in queue and have arc lengths $s_0 = 0, s_1, ..., s_k$ from the first of these points. And s_k

is the arc length of the current contact. The arc length between two adjacent points is approximated by their Euclidean distance, which is read by the robot.

We determine the next moving direction T_k of the sensor by extrapolating a polynomial fit to the tangential angles at the last k updates with arc length as the variable. Between the updates, the sensor moves in the same direction while sampling contacts. Shown below in Figure 2.3:

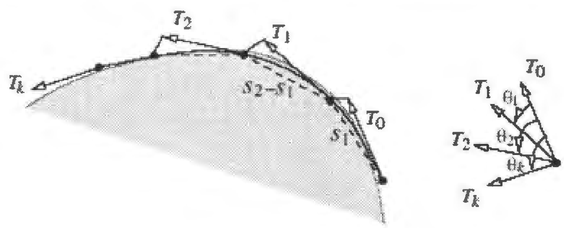


Figure 2.3 Tangent Direction History.

So $T_k = (\cos k, \sin k)$ will be the estimated tangent that determines the sensor motion until the next update. Next, T_k is added to the queue while T_0 is removed. Shown below in Figure 2.4:

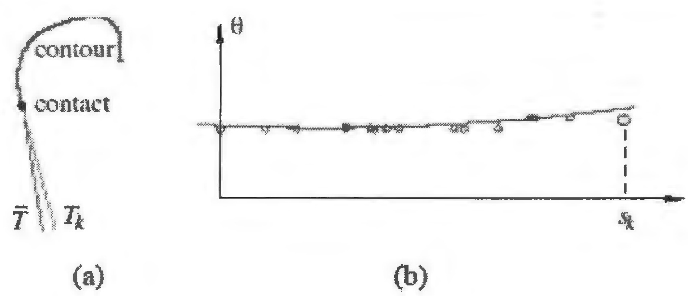


Figure 2.4 Prediction of the tracking motion.

2.2.2.2 Tracking Speed and Curvature Estimation

Curvature information is needed to slow down the speed while tracking a highly curved portion or a corner. As curvature of the contour increases, tracking needs to slow down to sample contour data more densely. Tracking around the corner on the contour in Figure 2.5 below:

Figure 2.6 compares the curvature function for the part of the real shape corresponding to the contour in Figure 2.4(a), and the set of curvature estimates got in tracking. Although the scales of arc

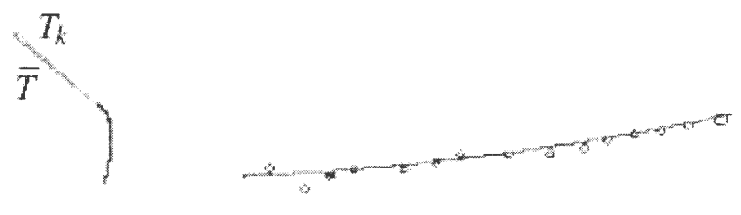


Figure 2.5 Tracking around the first corner.

length in the two diagrams are slightly different, we can still see the correspondences between the two major peaks:

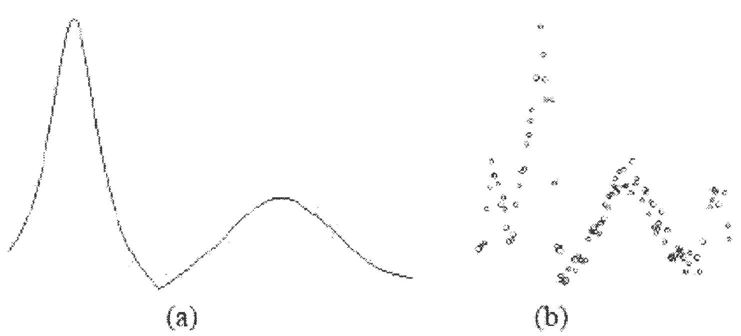


Figure 2.6 Estimating curvature on the fly.

2.2.3 Contour Adjustment

The contact between the shape and the joystick is estimated from the sensor's radius and the location of the Adept's end effectors. Due to the joystick bending, the projection of its upper end, which is attached to the end effectors is not tangent to the shape but rather intersects it. This is illustrated in Figure 2.7.

The larger the contact force, the more inside the shape boundary the estimated contact position. As a result, the generated contour would be slightly smaller than the original one without adjustment. In the meantime, the tracking motion also generates a zigzag contour, so some adjustment must be performed. The contour adjustment module is called upon to "grow" the contour outward based on force readings and reduce the zigzag effect. We pull every measured contact point outward along the

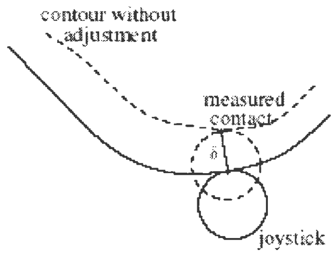


Figure 2.7 Contour adjustment.

normal direction. Shown below in Figure 2.8:

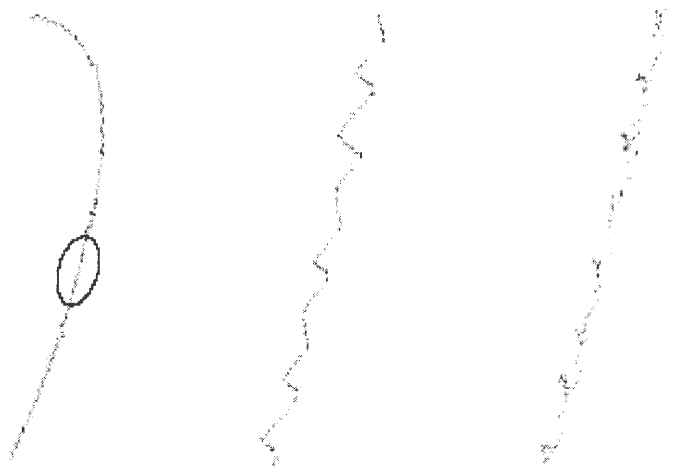


Figure 2.8 Contour adjustment results.

2.2.4 Curve Fitting with 3L Method

After we acquire large amount of raw data, we use 3L fitting method to describe the shape with one function in order to compress the storage and prepare for other further operation, comparisons, rotation, etc. In this section, we use the 3L algorithm (Implicit Polynomial Curves and Surfaces) to describe our shapes. With this method we can use arbitrary order of implicit polynomial with X, Y and Z. The cut on $Z = 0$ is the curve we need.

Let $f(x, y)$ be a d-order polynomial function of (x, y) given by:

$$\begin{aligned} f(x,y) &= \sum_{0\le i,j;0\le i+j\le d} a_{ij}x^i y^j \\ &= a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + \cdots + a_{0d}y^d \end{aligned} \tag{2.1}$$

The implicit polynomial curve representation for $f(x, y)$ is given by

$$z_f = \{(x,y) : f(x,y) = 0\} \text{where } z_f \text{ is called zero set of } f(x,y)$$

For detailed procedures of 3L algorithm, please reference Michael M. Blane, Zhibin Lei's paper [2].

Because our shapes are simple splines, a polynomial of order 4 or 5 is good enough to obtain nice descriptions. Below are the fitting results and comparisons between different orders.

2.3 Experiments

2.3.1 Tracking Results

Figure 2.9 displays the results of tracking a shape (a) at two different speeds. The contour in (b) was generated in time 16'30" with 20,328 data points while the contour in (c) in time 1'18" with only 1,170 data points. The two contours approximate the original shape with almost no difference can be seen.

(a) (b) (c)

Figure 2.9 Tracking at different speeds.

The results of tracking five more shapes are displayed in Figure 2.10. Comparisons between the contours and the original shapes are almost no difference.

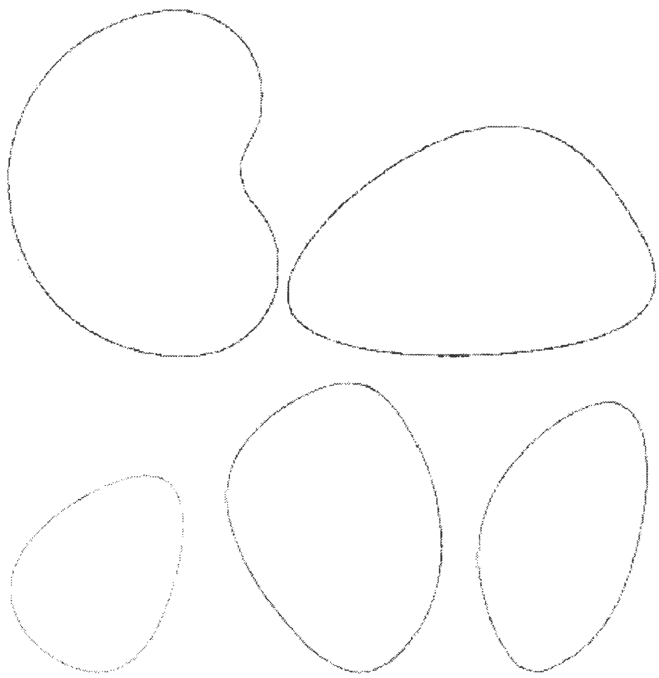


Figure 2.10 Contours of five tracked shapes.

2.3.2 Fitting Results

Since our shapes are simple splines, using 4 or 5 order polynomial is good enough to obtain nice descriptions. Below are the fitting results and comparisons between different orders.

We input the discrete data to the 3L algorithm and the results of different order polynomials are shown below. In each row, the first figure is the discrete data we get from the robot. In the second figure, we use order $d = 4$ to fit the data to a polynomial function. The third figure is order $d = 5$ and the fourth is order $d = 7$. The higher the order, the more noise presents in the fitting results. From our implementation, order $d = 4$ is the best, but for more complex shapes, we may need to use higher orders.

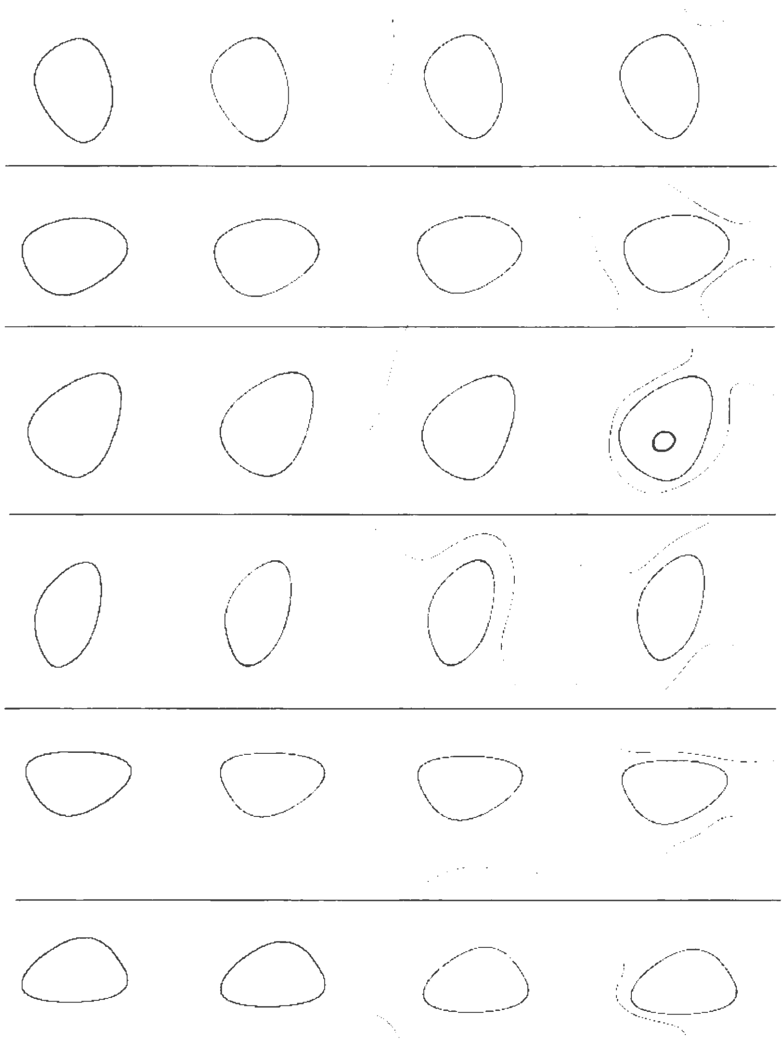


Figure 2.11 Different orders of fitting.

CHAPTER 3. RECONSTRUCTION OF 3D SURFACE PATCH

This section covers some basic concepts of surface patches for 3D object reconstruction, tactile sensor sampling strategy and reconstruction procedures for a single surface patch reconstruction.

3.1 Introduction

It is much more complex to reconstruct 3D shapes than those in 2D cases. This is not only simply due to just one more dimension. From sampling to post-processing, from model building to display are all involved the complexity of 3D computational geometry and also some computer graphics issues. The two most common approaches have been implemented in past decades are for medical imaging and range sensing, etc.

The first approach is using image processing techniques. Within this approach, a sequence of 2D cross section images is piled up to form a 3D image from up and downwards. This method costs much because of a large volume of image data. Interpolation will be implemented in neighborhood in the same coordinates frame. It is very precise to describe the details of anywhere in the model, but both storage and computation suffer from tens of thousand of images.

The second approach is to interpolate the boundaries of the cross section images. It also operates in the neighborhood, and gradually covers the boundaries of all the slices. Finally we can get a whole closed surface containing many basic shapes, e.g. triangles. A set of triangles are the basic components for 3D computer graphics.

Since the both the above methods require large amount of raw data to cover the whole shape, there is a demand for efficient geometry computation. As mentioned, tactile sensing resource is very limited, thus these two methods are hardly applicable with tactile sensors. To make use of tactile sensing, we at least have to guarantee the use of a very small amount of data. But to describe complex objects, limited data will result in huge error in final models. Therefore, we have to limit our tactile sensing data in a certain small local area of which the shape is not complex and can be described by limited data.

Our inspiration comes from the triangulation representation of a 3D surface in computer graphics, as the second approach described previously. We know that a pure triangulation suffers from large quantity and limited precision. Even the triangles are small enough to allow visualization on a computer screen but maybe too large for some high-end application. We cannot arbitrarily increase the number of triangles to reach a certain high precision. In computer graphics, triangles are the basic elements, but in our methodology, the atomic elements are small patches which makeup the whole surface. And more, each patch is described by a continuous polynomial functions within its own coordinates frame for robust and elastic manipulations with arbitrary precision.

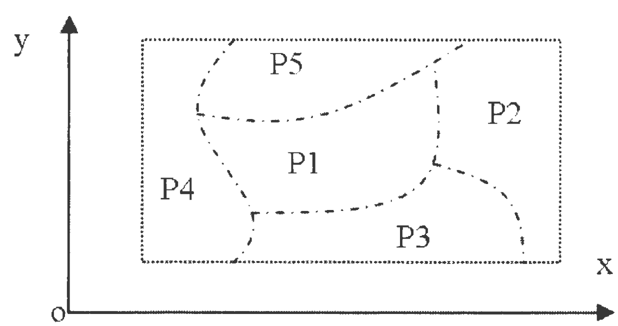


Figure 3.1 Regions of 3D surface.

In the simple example shown in Figure 3.1, it is a projection of a 3D surface on the X-Y plane. In the dotted rectangle area, it can be divided into 5 regions. Each region is described by a polynomial and the boundary which are represented within its own local coordinates frame, the transformation information between global coordinates frame and its own coordinates, and some other necessary information. Thus here, we have a list of 5 regions' description. To represent the whole surface of the rectangle area, we convert each region to the global coordinates and use global display framework. Each region has the freedom to adjust itself and a continuous function. In the global framework, we can transform and rotate each patch based on its information about pose and coordinates. In one word, we describe a 3D surface with a small number of continuous functions instead of thousands of fixed triangles.

In the first step, our attempt falls to dealing with a single patch which is a small local area. To divide the global surface into a local area problem has several advantages:

First, limiting the error in a small area to reduce the error of global computation. Second, reducing reconstruction efforts since anglicizing a small region is much easier than the whole shape. Third, it enhances the ability to describe a more complex surface with better local details other than describe the whole surface with just one function, e.g. 3L global fitting. Fourth, we can use the parameters of local geometry to increase robustness and reduce computation complexity.

Due to time limit, I just describe the reconstruction procedures for a single local patch. The organization of the rest of the chapter is as follows.

Section 3.2 introduces the basic strategy for single patch reconstruction. Section 3.3 covers the whole system architecture and data flows, including synthesis and actual experiments. Section 3.4 explains all theoretical implementation of the reconstruction process. Section 3.5 realizes the implementation of the above theorems and investigates the results with our 3D surface display environment. (*) Section **??** gives a summary of above work and outlines the future plans.

3.2 Strategy for A Single Patch

We carry out the steps to construct a single patch:

1. Sample along three curves on the surface along different cutting planes. These curves have a pre-selected common intersection point p.

2. After implementing the 2D-fitting on each curve in the corresponding cutting plane, get local geometry parameters at the common intersection point, such as tangent direction and local curvature.

3. Combine the tangent directions and curvatures from all the three curves, and calculate the surface normal direction and surface principle directions at the intersection point.

4. The surface principle directions and normal direction determin a new coordinates frame. Then convert the sample data from global coordinates to the new local coordinates.

5. Do 3D-Fitting on all original sample data within the new local coordinates frame and the resulting polynomial function is the representation of this surface patch.

Detailed information of each step will be described in later sections.

3.3 System Architecture and Data Flow

This section will give a global view of the system architecture and the complete data flow of the reconstruction procedures.

3.3.1 System Architecture

The whole system contains a synthesis simulation and also supports an interface with the real Cobra robot. Finally, the system will output a polynomial function. Then I display it in our 3D surface display environment. Thus, the simulated sampling and Cobra robot share a common input interface to the reconstruction algorithm. Figure 3.2 below shows the global system architecture:

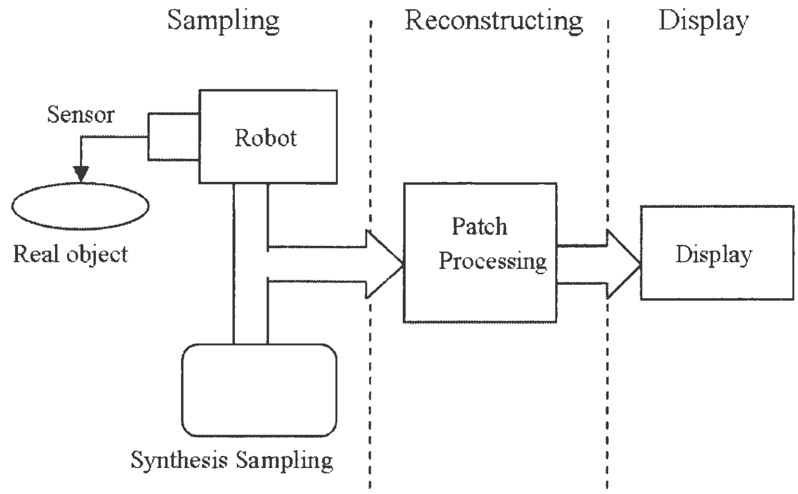


Figure 3.2 System architecture.

3.3.2 Data Flow

To describe the internal architecture of the Reconstruction Algorithm, I would like to use a data flow approach.

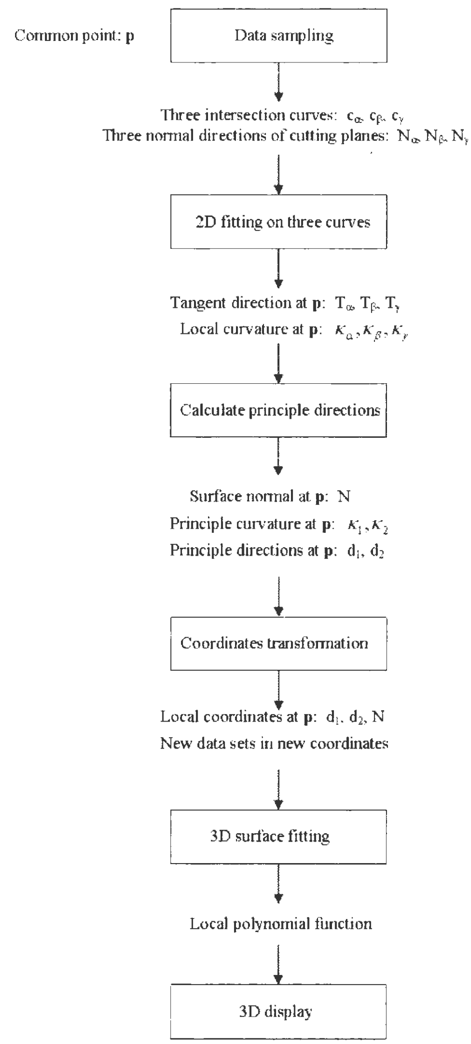


Figure 3.3 Data flow chart.

3.4 Reconstruction Procedures

3.4.1 Data Sampling

In order to execute an efficient data acquisition with very small amount of sampling data, we have to define a pattern to acquire the discrete data points. We cannot just randomly pick up points in the local area on the surface. Random points are totally unrelated with each other and we can not get relevant local geometry characters we want. In our approach, we introduce a *line sampling* method. Suppose a plane α determined by point p and the normal N_a intersects with a surface s and there is an intersection curve c_α on the surface. This curve c_α lies in the plane α. Our sample points are all from this intersection curve. From the view point inside the plane, the curve appears as a line on the surface and this why we call it line sampling. Below is the plot of this method:

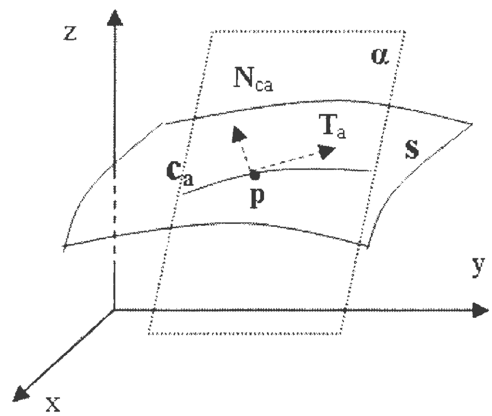


Figure 3.4 Line sampling.

Let $N_{c\alpha}$ be the normal direction of c_α at point p; T_α be the tangent direction of c_α at point p.

Such sampling strategy is used in both our synthesis implementation and real implementation with robot are following. The difference is synthesis data sampling supports arbitrary planes but real data sampling only supports planes whose normal directions lie in X-Y plane since the sensor is perpendicular to the X-Y plane of the work area.

We first choose a point p on the surface which is easy to reach in implementation. Then we choose a normal direction N_α for the cutting plane, and then within the plane, get sampling data along the curve as described above. Detailed sampling procedure will be discussed later. We can obtain three sets of sampling points if we choose three different plane normal directions. In order to make sure the p is the intersection point for the three curves; our Line Sampling always begins at p and extends to both sides with same number of sample points, shown as Figure 3.5.

Next two sub-sections introduce detailed operations for simulation purpose and real manipulation data from Cobra robot.

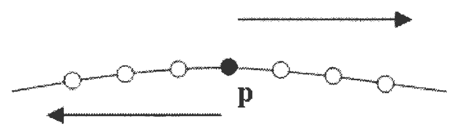


Figure 3.5 Sample points from p to both sides.

3.4.1.1 Simulation Sampling

The model we use to demonstrate the line sampling is ellipsoid with form:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{3.1}$$

Then we define the intersection point p as $(x_p, y_p, 0)$, the z value will be calculated during the sampling procedure.

The normal direction of the cutting plane is $n = (n_1, n_2, n_3)$, thus the plane is determined by:

$$((x, y, z) - p) \cdot n = 0 \tag{3.2}$$

From the equations above, we can obtain the function of the intersection curve of the ellipsoid and the cutting plane:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{((p - (x, y, 0) \cdot n)^2}{c^2 n_3^2} = 1 \tag{3.3}$$

For a given x value, we can find out the roots of above equation and the possible y values.

The sampling routine will begin at p. Let us define $\theta \approx \tan^{-1}(-n_1/n_2)$, for a fixed interval ds, the ith sample points pair will be

$$\begin{aligned} x_i &= x_p \pm i \cdot \cos\theta \cdot (ds) \\ y_i &= \text{positive root} \end{aligned} \tag{3.4}$$

Then we can get the corresponding z_i from Equation 3.1.

With pre-defined number num of sample points, the procedure will return a list of $2 * num + 1$.

We can run this routine with three different n values to get three Line Samplings.

3.4.1.2 Robot Sampling

The Cobra robot has a force sensor mounted at its arm end. This force sensor has a sharp needle end and can only detect the force along the z direction which is perpendicular with $X - Y$ plane. Below in Figure 3.6 is the architecture of this special designed sensor:

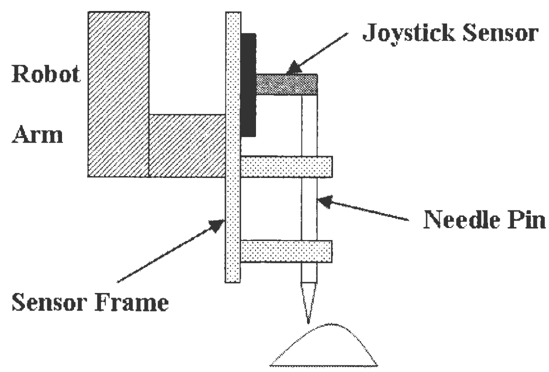


Figure 3.6 Sensor installation.

The arm first moves to $(x_0, y_0, ...)$, and then moves downwards slowly. When the pin makes contact with the object surface, it will be pushed up and the joystick will bend. Once the force falls into a valid range, the arm will stop and record this (x_0, y_0, z) point.

To do a *ling sampling*, the robot selects an angle α, and then move along the line determined by $(y - y_p) = \tan(\alpha) \cdot (x - x_p)$ in X-Y plane to sample $(2 \cdot num + 1)$ points:

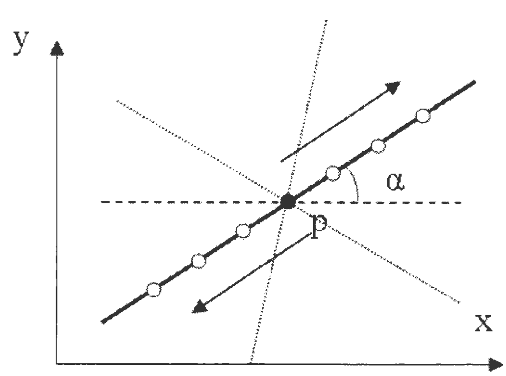


Figure 3.7 Sample from p to both sides.

Repeat above procedure, we can get three Line Samplings.

3.4.1.3 Output of Sampling Step

In this step, we obtain three sets of sample points along three intersection curves:

$$c_\alpha, c_\beta, c_\gamma$$

and corresponding normal direction of cutting planes:

$$N_\alpha, N_\beta, N_\gamma.$$

3.4.2 2D Fitting on Three Curves

Suppose we already get 3 sets of samples along three curves $c_\alpha, c_\beta, c_\gamma$ with a unique intersection point p. The next thing to do is to fit over these curves in cutting planes and getting curve tangent directions, surface normal direction and surface principle directions at point p.

To calculate the tangent direction of the intersection curve at point p, we will do least squares fitting to get a cubic polynomial function $f(t)$ in the three cutting plane. The tangent direction is:

$$T_\alpha = f'_\alpha(t) \tag{3.5}$$

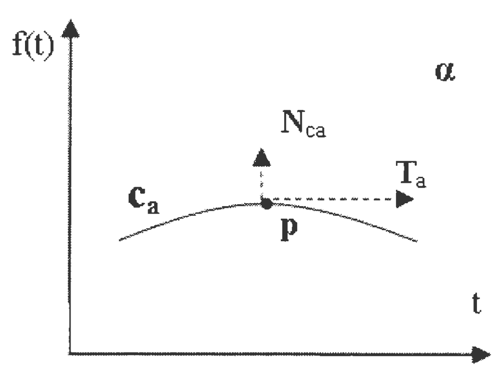


Figure 3.8 Fitting one curve in the cutting plane

Meanwhile we get the curvature κ_α as

$$\kappa = \frac{\| f' \times f'' \|}{\| f' \|^3} \tag{3.6}$$

In order to get more accurate curvature estimation, we only use points in a very near neighborhood, 3 or 4 from each side of p.

The output of this step includes tangent directions of the fitting curves at p

$$T_\alpha, T_\beta, T_\gamma$$

and local curvatures of the fitting curves at p

$$\kappa_\alpha, \kappa_\beta, \kappa_\gamma$$

.

3.4.3 Calculate Principle Directions

This step is the most important part of the whole processing, since the local coordinates are calculated now.

3.4.3.1 Surface Normal Direction at p

We know that $T_\alpha, T_\beta, T_\gamma$ are tangent directions of the intersection curves in cutting planes containing point p, thus these tangent directions must be in the tangent plane of the surface at point p.

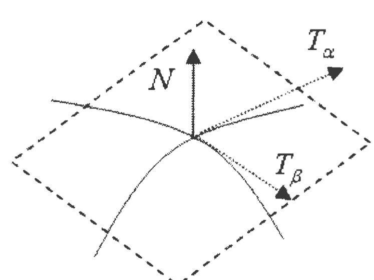


Figure 3.9 Surface normal direction N

The normal direction of the surface at point p is demoted by N and it is perpendicular to the tangent plane, thus T_α and T_β are perpendicular with N.

Therefore lead out the formula to calculate the surface normal:

$$N = \frac{T_\alpha \times T_\beta}{\| T_\alpha \times T_\beta \|} \tag{3.7}$$

We can make use of all the three tangent direction to reduce error by:

$$N = \frac{(T_\alpha - T_\lambda) \times (T_\beta - T_\gamma)}{\| (T_\alpha - T_\lambda) \times (T_\beta - T_\gamma) \|} \tag{3.8}$$

3.4.3.2 Principle Directions at p

Let φ_α be the angle between N and N_α, φ_β be the angle between N and N_β and φ_γ be the angle between N and N_γ. The normal curvatures in T_α, T_β and T_γ are $\kappa_\alpha \sin\varphi_\alpha$, $\kappa_\beta \sin\varphi_\beta$ and $\kappa_\gamma \sin\varphi$ respectively.

Let us see the T_α, T_β and T_γ in the tangent plane:

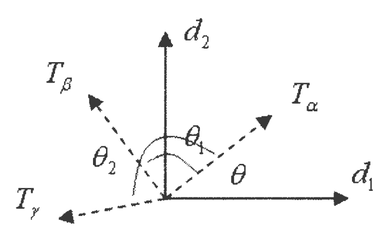


Figure 3.10 Calculation of principle directions

In the Figure 3.10, d_1 and d_2 are two principle directions, and $d_1 \perp d_2$. Here, θ is the angle between d_1 and T_α which is unknown; θ_1 is the angle between T_α and T_β, θ_2 is the angle between T_α and T_γ. Both θ_1 and θ_2 can be calculated since T_α, T_β and T_γ are known.

From the properties of principle curvatures, we can obtain below relations:

κ_1 is the principal curvature along d_1; κ_2 is the principal curvature along d_2.

$$\begin{aligned} \kappa_\alpha \sin\varphi_\alpha &= \kappa'_\alpha = \kappa_1 \cos^2\theta + \kappa_2 \sin^2\theta \\ \kappa_\beta \sin\varphi_\beta &= \kappa'_\beta = \kappa_1 \cos^2(\theta + \theta_1) + \kappa_2 \sin^2(\theta + \theta_1) \\ \kappa_\gamma \sin\varphi_\gamma &= \kappa'_\phi = \kappa_1 \cos^2(\theta + \theta_2) + \kappa_2 \sin^2(\theta + \theta_2) \end{aligned} \tag{3.9}$$

Here $\kappa_\alpha, \kappa_\beta, \kappa_\gamma$ can be obtained from the 2D curve fitting. $\varphi_\alpha, \varphi_\beta, \varphi_\gamma$ are the angles between surface normal directions and the plane normal directions. θ_1, θ_2 are the angles of (T_α - T_β) and (T_α - T_γ). Thus within this system, $\theta_1, \theta_2 \kappa_\alpha, \kappa_\beta, \kappa_\gamma$, $\varphi_\alpha, \varphi_\beta, \varphi_\gamma$, θ_1, θ_2 are known, only κ_1, κ_2 and θ are unknown. By solving these equations, we can obtain the principle directions and principle curvatures.

The solution of θ is:

$$tg(2\theta + \theta_2) = \frac{\sin(\theta_1 - \theta_2)}{\frac{\kappa'_\alpha - \kappa'_\beta}{\kappa'_\alpha - \kappa'_\gamma} \cdot \frac{\sin\theta_2}{\sin\theta_1} \cdot \cos(\theta_1 - \theta_2)} \tag{3.10}$$

When we get the θ value, put it back into the above formulas. With any two of them, we can get the solution of that linear system, κ_1 and κ_2.

To obtain principle direction d_1, just rotate the T_α clockwise by θ; then rotate d_1 counterclockwise by $\pi/2$ to get d_2.

3.4.3.3 Outputs

After this procedure, we will get surface normal direction at $p : N$; principle directions d_1 and d_2; principle curvatures κ_1 and κ_2.

3.4.4 Coordinates Transformation

Why we select d_1, d_2 and N as our local coordinates?

1. The x-axis and y-axis are the principle directions

2. The normal curvatures along x-axis and y-axis are principle curvatures

3. In further fitting function, the x, y and xy terms are eliminated.

In order to fit in this local coordinates frame, we need to convert the coordinates of original sample data.

For a point $q = (x, y, z)$ in the original coordinates, and its location in new coordinates frame is $q' = (x', y', z')$, so

$$q = p + x'\vec{d}_1 + y'\vec{d}_2 + z'\vec{N} \tag{3.11}$$

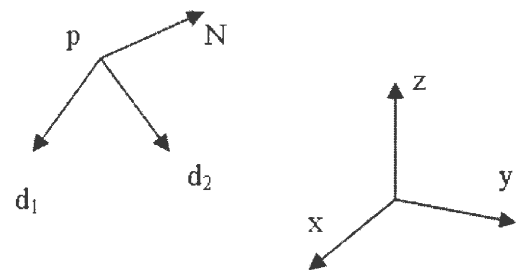


Figure 3.11 Coordinates transformation

where,

$$\begin{aligned} x' &= (q-p)\cdot \vec{d}_1 \\ y' &= (q-p)\cdot \vec{d}_2 \\ z' &= (q-p)\cdot \vec{N}, \end{aligned} \tag{3.12}$$

thus the above implies

$$\begin{aligned} q &= p + (\vec{d}_1\ \vec{d}_2\ \vec{N})q' \\ q' &= (\vec{d}_1\ \vec{d}_2\ \vec{N})^T(q-p) \end{aligned} \tag{3.13}$$

Then we transform all original data points to new local coordinates.

3.4.5 3D Surface Fitting

The surface fitting in the new coordinates frame here is very simple, similar with 2D-fitting mentioned previously. We just input all the transformed sample points into the fitting method. The fitting function is given as:

$$z(x,y) = \frac{1}{2}k_1x^2 + \frac{1}{2}k_2y^2 + \sum_{i+j\geq 3} a_{i,j}x^iy^j \tag{3.14}$$

In our implements, we choose order 4. Then we solve the least square fitting problem.

The fitting function

$$F(x_i) = c_1\phi_1(x_i) + c_2\phi_2(x_i) + \cdots + c_k\phi_k(x_i)$$
$$\text{to satisfy} \tag{3.15}$$
$$\sum_{i=1}^{n} (f_i - F(x_i))\phi_j(x_i) = 0, j = 1, \cdots, k$$

leads to a problem

$$Pc = q,$$

$$\text{P is a } k \times k \text{ matrix}$$
$$\mathrm{p_{lj}} = \phi_l^T \phi_j = \sum_{i=1}^{n} \phi_l(x_i)\phi_j(x_i) \tag{3.16}$$

$$q \text{ is a } k \times 1 \text{ matrix}$$
$$q_l = f^T \phi_l = \sum_{i=1}^{n} f_i\phi_l(x_i)$$

and $c = (c_1, c_2, \cdots, c_k)$ is the solution.

In our application, the basis functions are chosen as follows:

$$\phi_1 = x^3$$
$$\phi_2 = x^2y^1$$
$$\phi_3 = x^1y^2$$
$$\phi_4 = y^3$$
$$\phi_5 = x^4$$
$$\phi_6 = x^3y$$
$$\phi_7 = x^2y^2$$
$$\phi_8 = x^1y^3$$
$$\phi_9 = y^4$$

Using all these items will lead to unstable results since the items containing odd order cannot be bounded between the sampling curves. In our experiments, we assume the area is local enough and then we can just use x^4, x^2y^2, y^4 to fit.

3.4.6 3D Display

At the very beginning, we are using Mathematica and Matlab to display the 3D surface, both simulation and final results. But the display from them cannot be integrated in my own program. Thus I have to feed the data into Mathematica and Matlab every time I do a new experiment. Though I wrote a translator to generate the codes for those mathematic tools, it is still very inconvenient. What is more, we cannot rotate or transform the 3D surface freely. Due to these drawbacks, I finally developed my own 3D display environment with OpenGL and GLUT utility.

My display supports any formulas in the forms of

$$f(x, y) = g_1(x, y) + g_2(x, y) + \cdots \qquad g_i(x, y)\ could\ be\ any\ functions.$$

or

$$F(x, y, z) = 0.$$

Its features include:

1. Automatic display scaling and size adjustment
2. Free rotating with keyboard or USB game controller
3. Automatic rotating
4. Color/Gray mapping of z values, can be customized
5. Can be embedded in any windows with modified GLUT library

Here are some display examples in Figure 3.12 and Figure 3.13. The display of fitting results in previous sub-section will be discussed in next section.

3.5 Implementation Results

3.5.1 Simulation Results

All synthesis experiments are done with ellipsoid

$$\frac{x^2}{8^2} + \frac{y^2}{5^2} + \frac{z^2}{3^2} = 1.$$

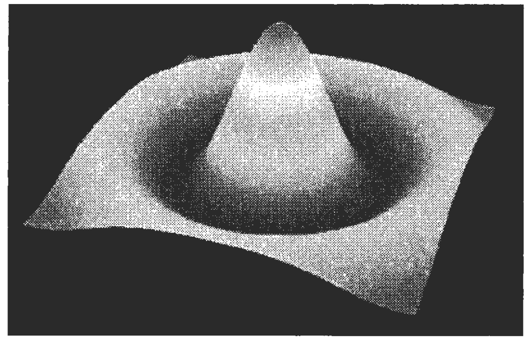


Figure 3.12 Display Example 1: $z = 6*\sin(\sqrt{x^2+y^2}+0.001)/(\sqrt{x^2+y^2}+0.001)$

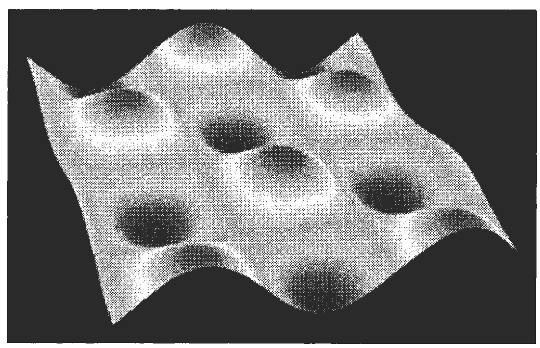


Figure 3.13 Display Example 2: $z = \sin(x) * \cos(y)$

To estimate the principle directions and curvatures, we first select a point p, for example $(1, 2, *)$, then select three normal direction to define three cutting planes. Then we execute the steps previously described.

3.5.1.1 Results of Principle Directions and Principle Curvatures

To compare the results of principle curvatures, we repeat the procedures on several points $p_1, p_2, \cdots$ on the surface of the ellipsoid along the x axis shown below in Figure 3.14

hen we draw the estimated principle curvatures and actual principle curvatures in a plot 3.15: Blue dots are simulation results and red dots are actual values.

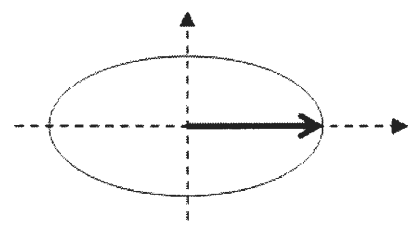


Figure 3.14 Estimate curvatures along x-axis

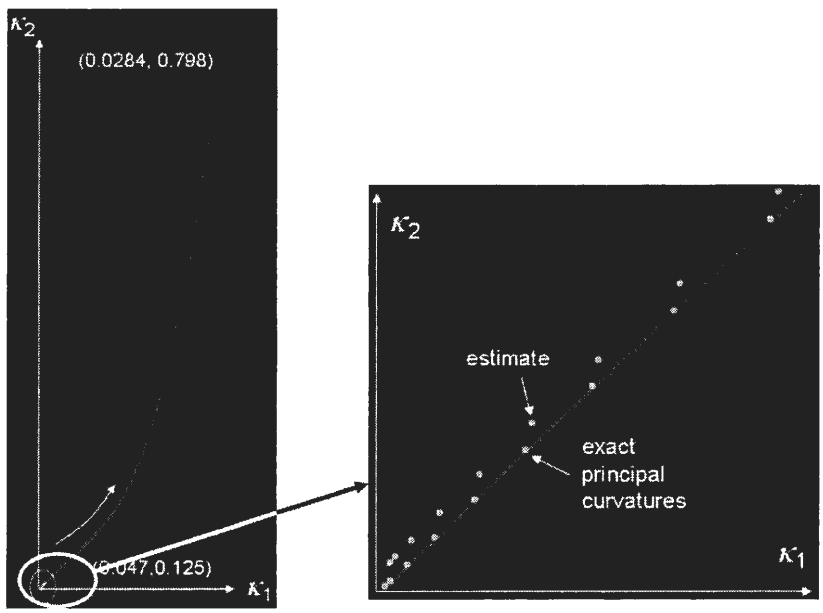


Figure 3.15 Compare estimated curvatures and actual values

3.5.1.2 Results of 3D Fitting in Display

We select an arbitrary point $p = (2, 1, 0)$ then reconstruct the local area around it on the surface of the ellipsoid.

Principle Directions: $(0.995, -0.0455, -0.0927)$ and $(0.0333, 0.991. - 0.129)$.

Principle Curvatures: -0.051678 and -0.129094.

The fitting polynomial function is display with z-height color in Figure 3.16.

Next, we evaluate the Z values of the fitting functions and compare them with the sampling values to verify the error of the function, in Figure 3.17.

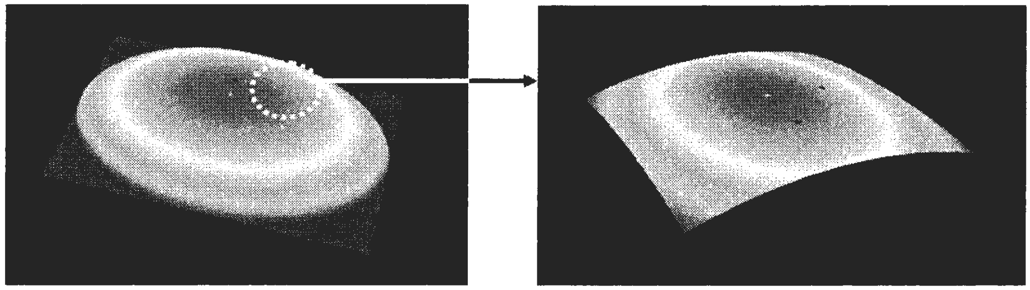


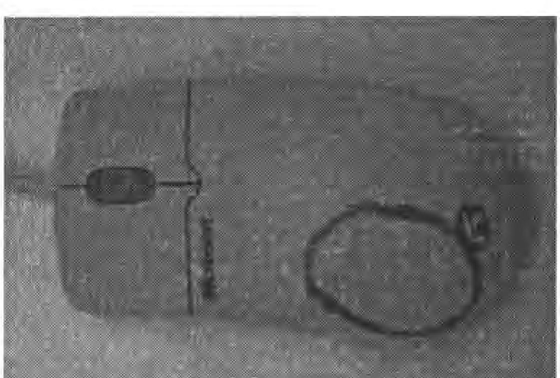
Figure 3.16 3D display of fitting at (2,1,0)

3.5.2 Robot Manipulation Results

With Cobra robot and the joystick sensor, we do sampling on the surface of a mouse. The original mouse and the fitting result of a small patch one its surface is compared in Figure 3.18.

Z value of sample points	Z value of fitting results
-0.018155163441274	-0.0186285743761176
-0.0139553828864855	-0.0142650742814843
-0.0102961831937466	-0.0104833982368479
-0.0071825554809217	-0.00728290998211672
-0.00461986596754422	-0.00466331027386279
-0.00261387607781557	-0.0026246459014693
-0.00117075732095257	-0.00116731187662577
-0.000297113113 68522	-0.0002920591521165
0	0
-0.000286951569103067	-0.000292614261292609
-0.00116600439867085	-0.00117175561657926
-0.00264572744754303	-0.00263965907655316
-0.00473525218780566	-0.00469894677536798
-0.00744430906016882	-0.00735263735878757
-0.0107832641171251	-0.0106041525722868
-0.0147631603553121	-0.0144573252108608
-0.0193957649925539	-0.0189164092195245

Figure 3.17 Error comparisons in new coordinates



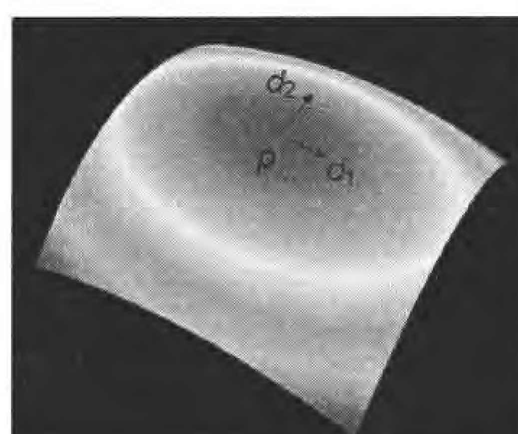


Figure 3.18 Reconstruction a patch of a mouse surface

CHAPTER 4. SUMMARY AND DISCUSSION

4.1 2D Tracking

We have described a hybrid force/position control system that tracks curved contours very precisely with a simple joystick sensor. Experiments have demonstrated high quality of tracking. Curvature is estimated at the same time of tracking, which allows dynamic control of the tracking speed based on local geometry. Latencies in robot acceleration and deceleration are obstacles to the tracking speed. Communication delays between the PC and the sensor and between the PC and the robot also slow down the tracking. Extension to 3D surface tracking lies in reliable estimation of surface normal (i.e., the tangent plane at contact). We may try to use tactile array sensors for the purpose. But some additional force sensor may be needed to carry out force control in the future. The fitting works well current, but we also need to find out a better solution for very complicated shapes which means there must be a method to filter the noises come from high order polynomials.

4.2 3D Patches

We used a simple method to sample the position data at a certain point on the object surface. Then we showed a method to reconstruct a surface patch with three sample lines. All the sample lines shared a common intersection point. With Least-Square fitting algorithm, we could get the local geometry characters of each curve(sample line). Then we calculated the principle directions at the intersected point. Based on these new principle directions, we did 3D surface fitting in a new coordinates framework. We got good results from simulations and actual experiments with mouse, cylinder, etc. This method is very simple and could get fairly good surface models without array sensors. But it takes more time to accomplish an experiment since it gets data point by point while array sensor can get them at the same time. In the future, we would like to increase the sampling speed and also increase the robustness. Also, we are very interested in how to join multiple patched together in a certain neighborhood.

BIBLIOGRAPHY

[1] P. K. Allen and P. Michelman. Acquisition and interpretation of 3-D sensor data from touch. *IEEE Trans. Robot. & Automat.*, 6(4):397–404, 1990.

[2] J. Baeten and J. De Schutter. Hybrid vision/force control at corners in planar robotic-contour following. *IEEE/ASME Trans. Mechatronics*, 7(2):143-151, 2002.

[3] M. M. Blane. The 3L algorithm for fitting implicit polynomial curves and surfaces to data. *IEEE Trans. Pattern Analys. & Mach. Intell.*, 22(3):298-313, 2000.

[4] R. E. Ellis and M. Qin. Singular-value and finite-element analysis of tactile shape recognition. In *Proc. IEEE Int. Conf. Robot. & Automat.*, pp. 2529–2535, 1994.

[5] R. S. Fearing. Tactile sensing for shape interpretation. In S. T. Venkataraman and T. Iberall, eds., *Dextrous Robot Hands*, pp. 209–238. Springer-Verlag, 1990.

[6] F. Jatta, G. Legnani, and A. Visioli. Hybrid force/velocity control of industrial manipulators with elastic transmissions. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Sys.*, pp. 3276-3281, 2003.

[7] Y.-B. Jia. Contact sensing for parts localization: sensor design and experiments. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Sys.*, pp. 516-522, 2003.

[8] O. Khatib and J. Burdick. Motion and force control of robot manipulators. In *Proc. IEEE Intl. Conf. Robot. & Automat.*, pp. 1381-1386, 1996.

[9] F. Lange and G. Hirzinger. Learning force control with position controlled robots. In *Proc. IEEE Intl. Conf. Robot. & Automat.*, 1996.

[10] M. T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Trans. Systems, Man, & Cybernetics.*, 11(6):418-432, 1981.

[11] M. Moll and M. A. Erdmann. Reconstructing the shape and motion of unknown objects with active tactile sensors. In J.-D. Boissonnat et al., eds., *Algorithmic Foundations of Robotics.*, V, pp. 293-309. Springer-Verlag, 2004.

[12] M. H. Raibert and J. J. Craig. Hybrid position/force control of manipulators. J. Dynamic Systems, *Measurement, & Control*, Trans. of ASME, 102:126-133, 1981.

[13] D. Xiao, B. K. Ghosh, N. Xi, and T. J. Tarn. Sensor-based hybrid position/force control of a robot manipulator in an uncalibrated environment. *IEEE Trans. Control Sys. Tech.*, 8(4):635-645, 2000.

[14] T. Yoshikawa and A. Sudou. Dynamic hybrid position/force control of robot manipulators-on-line estimation of unknown constraint. *IEEE Trans. Robot. & Automat.*, 9(2):220-226, 1993.

ACKNOWLEDGMENTS

I would like to thank my advisor, Yan-Bin Jia, for his guidance, help and support throughout this research. I also would like to thank my committee members David Fernandez-Baca and Greg R. Luecke for their help in completing this work.

Support for this research has been provided in part by Iowa State University, and in part by the National Science Foundation through a CAREER award IIS-0133681.